实验一 C++程序设计基础

一、 实验目的

1、 了解和使用 C++开发环境

2、 了解基本的控制台项目的创建、编译和运行

3、 了解 Visual Studio 的编辑功能、项目配置属性

4、 掌握 C++开发中的调试功能、调试菜单、错误窗口

5、 了解 std::vector 基本功能

二、 实验内容

使用 Visual Studio 环境来编辑、编译和运行下列程序。

//代码内容 1

```
#include <iostream>
using namespace std;

enum GameResult { WIN, LOSE, TIE, CANCEL };

int main() {
    GameResult  result;                         //声明变量时,可以不写关键字 enum
    enum GameResult omit = CANCEL;          //也可以在类型名前写 enum

    for (int count = WIN ; count <= CANCEL ; count++) {//隐含类型转换
        result = GameResult(count);           //显式类型转换
        if (result == omit)
            cout << "The game was cancelled" << endl;
        else {
            cout << "The game was played ";
            if (result == WIN)
                cout << "and we won!";
            if (result == LOSE)
                cout << "and we lost.";
            cout << endl;
        }
    }
```

```
    return 0;
}
```

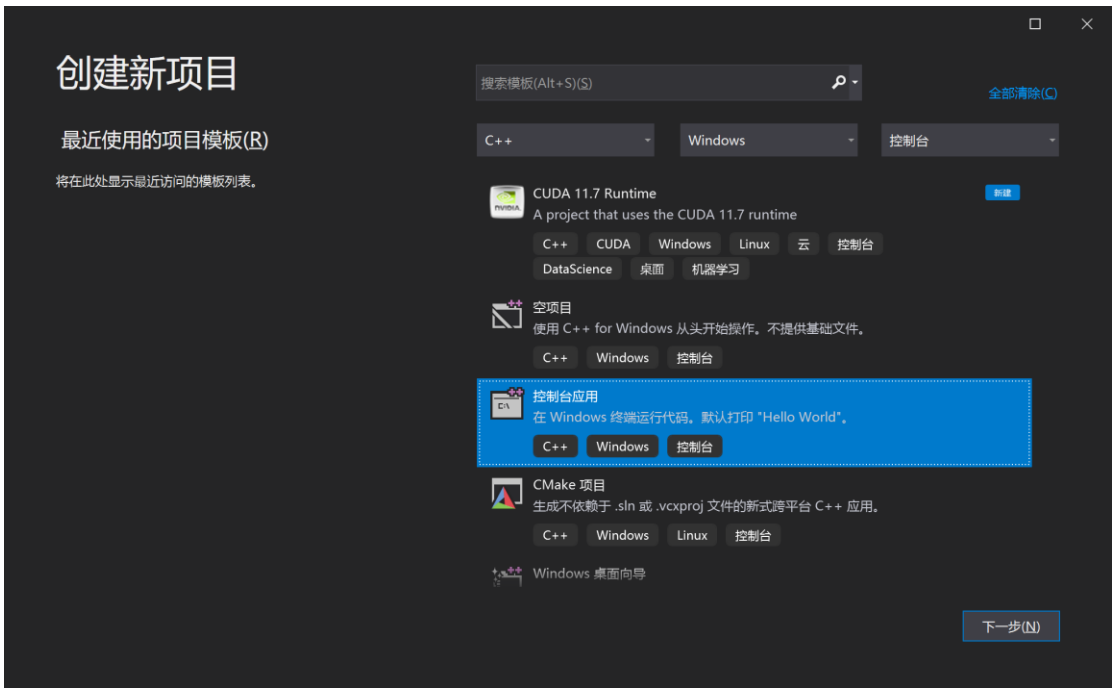
//代码内容 2

```
#include <iostream>
#include <stdlib.h>
#include <vector>
using namespace std;

int main() {
    const int N = 4;
    for (int i = 1; i <= N; i++) {//输出前 4 行图案
        for (int j = 1; j <= 30; j++)
            cout << ' ';   //在图案左侧空 30 列
        for (int j = 1; j <= 8 - 2 * i; j++)
            cout << ' ';
        for (int j = 1; j <= 2 * i - 1; j++)
            cout << '*';
        cout << endl;
    }
    for (int i = 1; i <= N - 1; i++) { //输出后 3 行图案
        for (int j = 1; j <= 30; j++)
            cout << ' ';   //在图案左侧空 30 列
        for (int j = 1; j <= 7 - 2 * i ;j++)
            cout << '*';
        cout << endl;
    }

    std::vector<int > a = { 1,2,3 };
    std::cout << a[4] << std::endl;

    return 0;
}
```

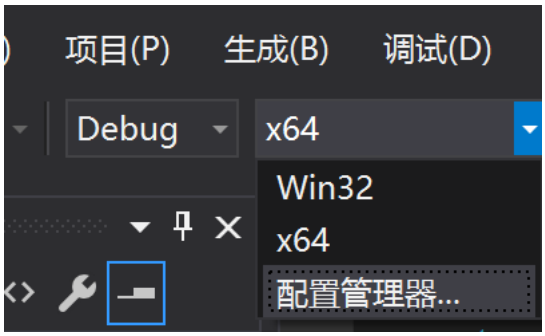
三、 实验步骤

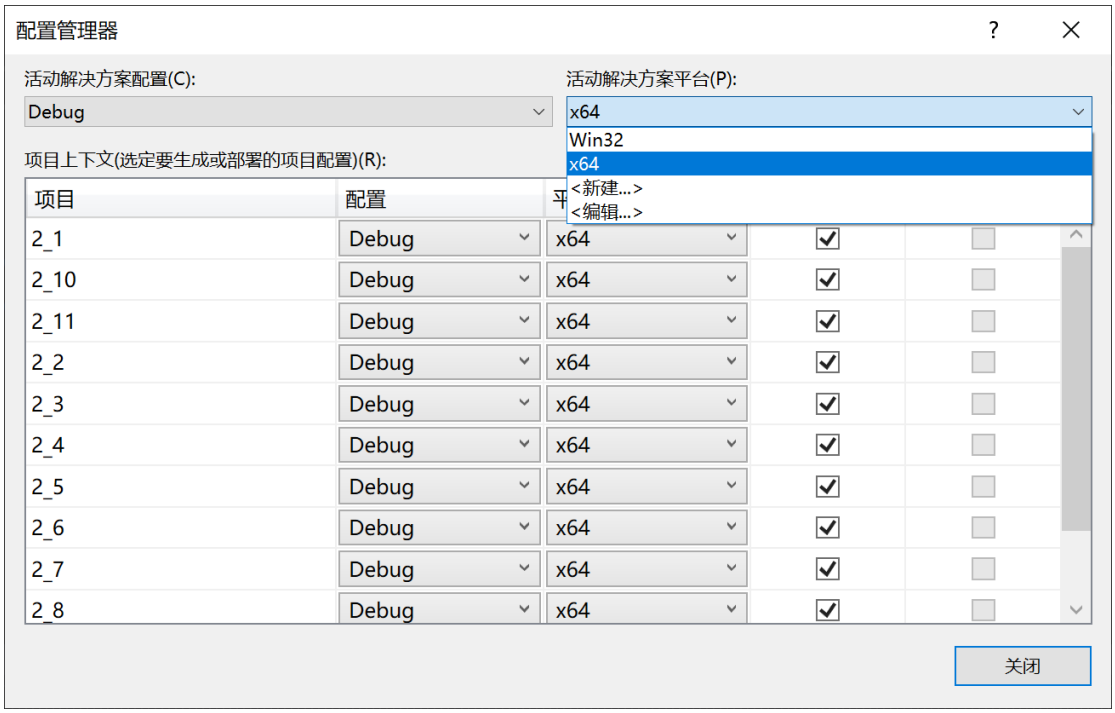
1、选择文件-新建-项目

2、选择 Windows 平台-控制台应用

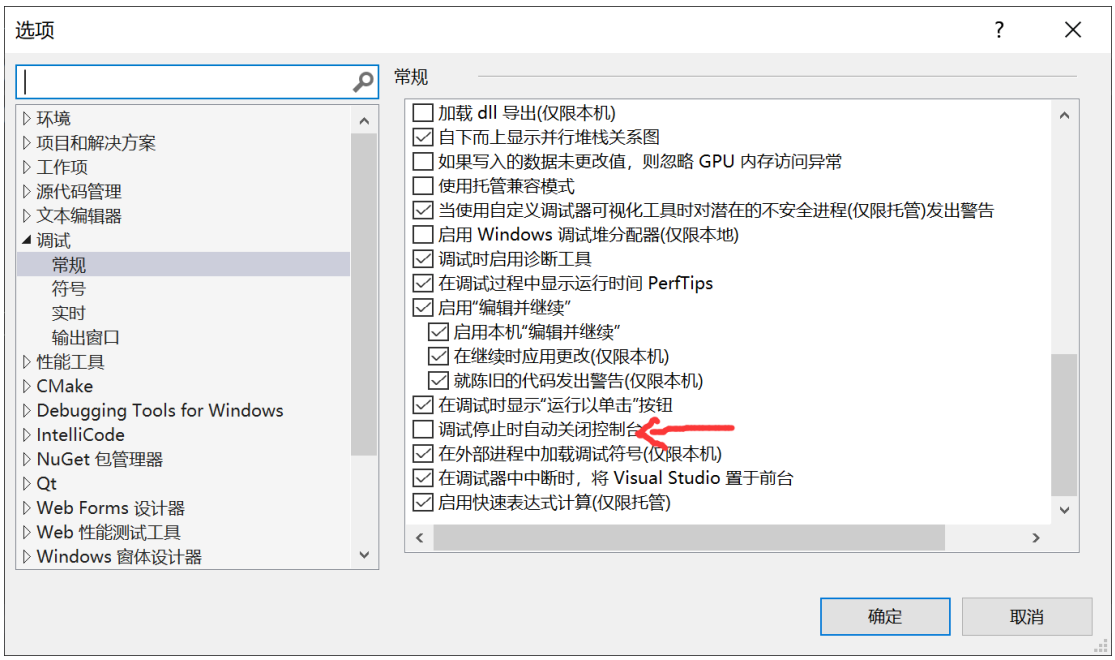


6、 输入本实验的代码内容。选择编译平台为 x64

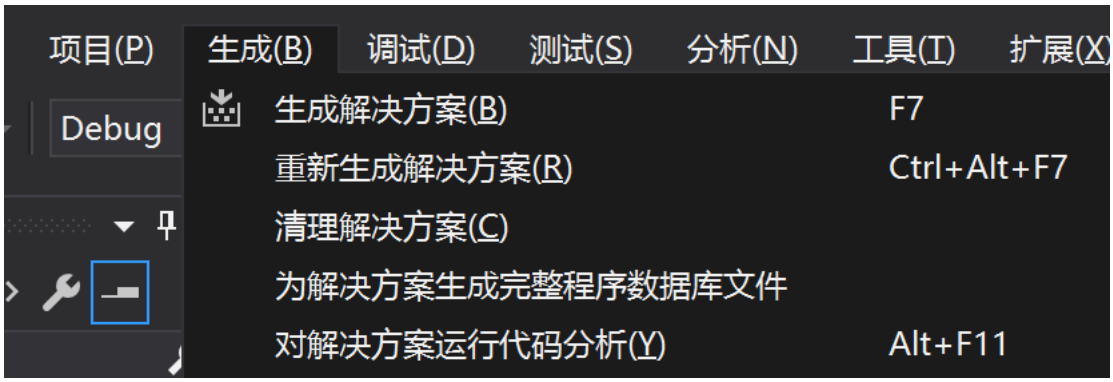




7、 点击工具-选项，去掉“调试停止时自动关闭控制台”

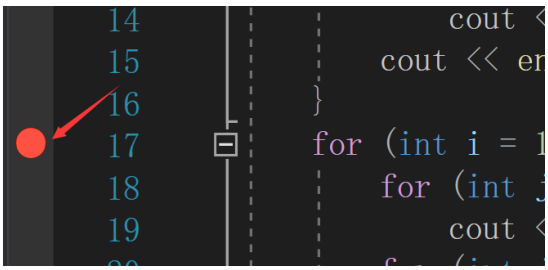


6、点击运行-生成解决方案或者重新生成解决方案

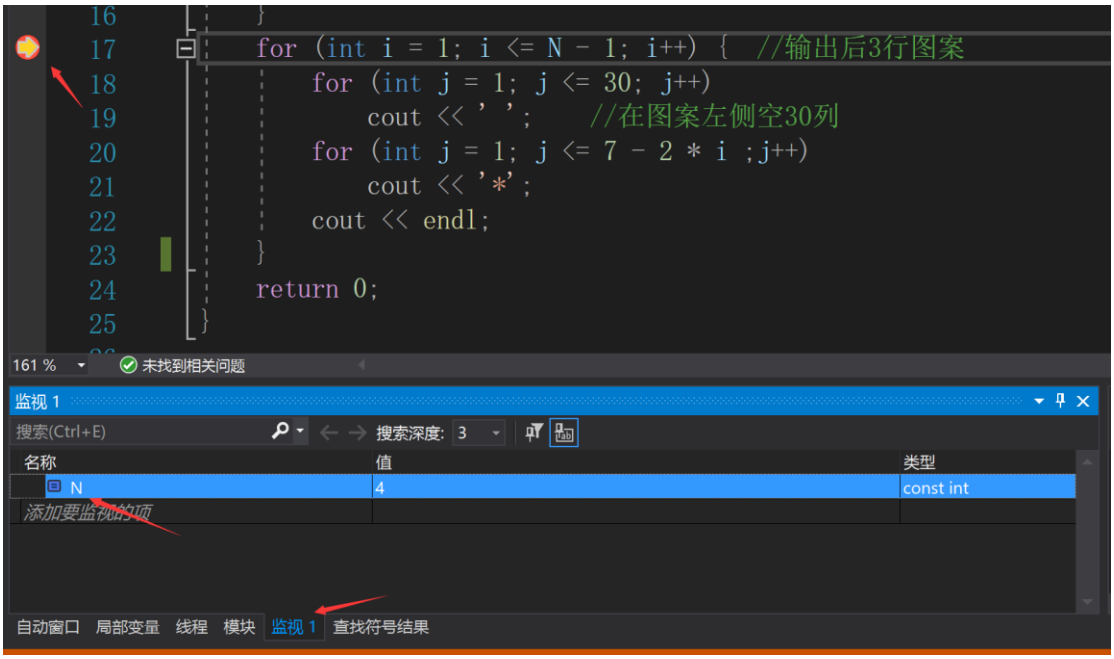


8、 点击运行按钮，运行程序

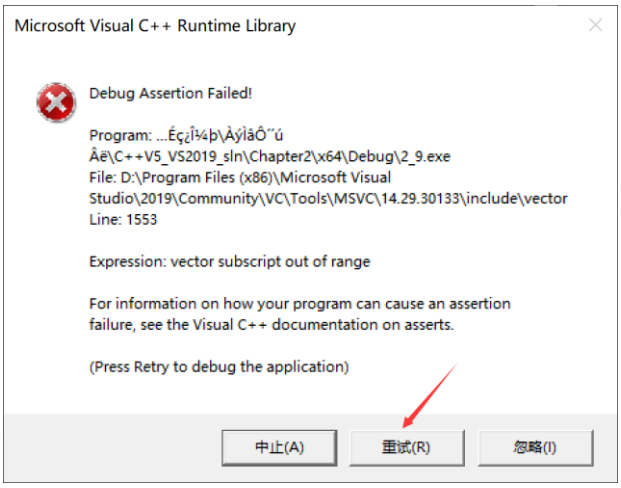
9、 在左侧单击，生成红色圆点



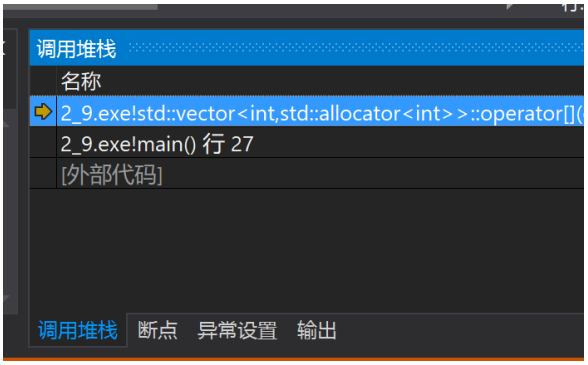
10、 点击运行按钮，将会在圆点处停止。下方监视窗口，可以双击进入编辑状态，然后输入需要查看的变量。



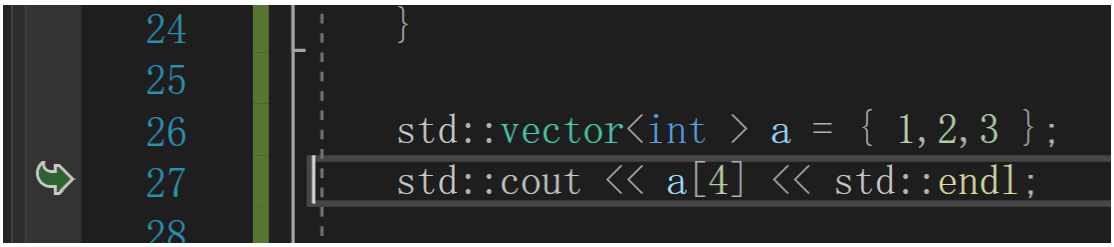
11、 输入代码内容 2，先进入 Release 模式运行，然后进入 debug 模式运行后，弹出报错窗口，点击重试。



12、 打开调用堆栈窗口，找到最近的我们自己编写的代码，并双击，就定位到错误代码地方。



13、 我们发现这里越界了，修改代码后恢复正常。



四、 Std::vector 讲解

在 C++ 中，std::vector 是标准库中最常用的容器之一，它可以根据需要自动扩容，能够高效地插入、删除和访问元素。本文将深入介绍 std::vector 的使用方法，包括创建、初始化、插入、删除、访问元素等方面。

1、创建和初始化

std::vector 是一个动态数组容器，可以存储同一类型的元素，例如整数、浮点数、字符串等。下面是一个创建和初始化 std::vector 对象的示例代码：

```
#include <vector>
#include <iostream>
int main() {
    // 创建一个空的 vector 对象
    std::vector<int> myVector;
    // 创建一个含有 5 个元素的 vector 对象
    std::vector<int> myVector2(5);
    // 创建一个含有 5 个元素，每个元素都是 10 的 vector 对象
    std::vector<int> myVector3(5, 10);
    // 创建一个 vector 对象，初始化为另一个 vector 对象的副本
    std::vector<int> myVector4(myVector3);
    // 输出 myVector3 中的元素
    std::cout << "myVector3 中的元素是： " << std::endl;
    for (int i = 0; i < myVector3.size(); ++i) {
        std::cout << myVector3[i] << " ";
    }
    std::cout << std::endl;
```

```
    return 0;
}
```

在上面的代码中，我们创建了四个不同的 std::vector 对象，并分别进行了初始化。第一个 std::vector 对象是一个空的对象，不包含任何元素。第二个 std::vector 对象是含有 5 个元素的对象，但没有指定元素的值，默认初始化为 0。第三个 std::vector 对象是含有 5 个元素的对象，每个元素都是 10。第四个 std::vector 对象是初始化为第三个 std::vector 对象的副本。

2、插入和删除元素

std::vector 对象可以动态增加和删除元素，我们可以使用 push_back() 函数向 std::vector 对象中添加元素。下面是一个示例代码，展示了如何向 std::vector 对象中添加元素：

```
#include <vector>
#include <iostream>
int main() {
    // 创建一个空的 vector 对象
    std::vector<int> myVector;
    // 向 vector 中添加元素
    myVector.push_back(1);
    myVector.push_back(2);
    myVector.push_back(3);
    // 输出 vector 中的元素
    std::cout << "vector 中的元素是： " << std::endl;
    for (int i = 0; i < myVector.size(); ++i) {
        std::cout << myVector[i] << " ";
    }
    std::cout << std::endl;
    // 删除 vector 中最后一个元素
    myVector.pop_back();
    // 输出 vector 中的元素
    std::cout << "vector 中的元素是： " << std::endl;
    for (int i = 0; i < myVector.size(); ++i) {
        std::cout << myVector[i] << " ";
    }
    std::cout << std::endl;
    return 0;
}
```

3、访问元素

std::vector 对象可以像数组一样访问其元素。我们可以使用下标运算符（[]）或 at()函数来访问 std::vector 对象中的元素。下面是一个示例代码，展示了如何访问 std::vector 对象中的元素：

```
#include <vector>
#include <iostream>
 int main() {
```

```
        // 创建一个含有 3 个元素的 vector 对象
        std::vector<int> myVector(3);
        myVector[0] = 1;
        myVector[1] = 2;
        myVector[2] = 3;
        // 使用下标运算符访问 vector 中的元素
        std::cout << "vector 中的第一个元素是: " << myVector[0] << std::endl;
        // 使用 at() 函数访问 vector 中的元素
        std::cout << "vector 中的第二个元素是: " << myVector.at(1) << std::endl;
        return 0;
    }
```

4、其他操作

size()：返回 std::vector 对象中元素的个数。
capacity()：返回 std::vector 对象中已分配的存储空间的大小。
reserve(n)：为 std::vector 对象分配至少能容纳 n 个元素的存储空间。
clear()：从 std::vector 对象中删除所有元素。
empty()：判断 std::vector 对象是否为空。
sort: 对 std::vector 进行排序
insert：在 std::vector 中间插入数据
pop_back: 删除末尾数据
erase：删除中间数据。
find：查找数据

```
#include <vector>
#include <iostream>
#include <algorithm>

int main() {
    // 创建一个含有 3 个元素的 vector 对象
    std::vector<int> myVector = { 3,4,1,2,7,6 };
    // 输出 vector 中元素的个数和已分配的存储空间的大小
    std::cout << "vector 中元素的个数为: " << myVector.size() << std::endl;
    std::cout << "vector 中已分配的存储空间的大小为: " << myVector.capacity() << std::endl;
    // 为 vector 对象分配至少能容纳 10 个元素的存储空间
    myVector.reserve(10);
    // 输出 vector 中元素的个数和已分配的存储空间的大小
    std::cout << "vector 中元素的个数为: " << myVector.size() << std::endl;
    std::cout << "vector 中已分配的存储空间的大小为: " << myVector.capacity() << std::endl;
    // 删除 vector 中的所有元素
```

```
        myVector.clear();
        // 输出 vector 中元素的个数和已分配的存储空间的大小
        std::cout << "vector 中元素的个数为: " << myVector.size() << std::endl;
        std::cout << "vector 中已分配的存储空间的大小为: " << myVector.capacity() << std::endl;
        // 判断 vector 是否为空
        if (myVector.empty()) {
            std::cout << "vector 为空。" << std::endl;
        }
        else {
            std::cout << "vector 不为空。" << std::endl;
        }
        myVector = std::vector<int>({ 3,4,1,2,7,6 });
        std::cout << "vector 排序 1。" << std::endl;
        std::sort(myVector.begin(), myVector.begin()+4);
        for (auto& it : myVector) {
            std::cout << it << " ";
        }
        std::cout << std::endl;
        std::cout << "vector 排序 2。" << std::endl;
        std::sort(myVector.begin(), myVector.end());
        for (auto& it : myVector) {
            std::cout << it << " ";
        }
        std::cout << std::endl;
        std::cout << "vector 插入。" << std::endl;
        myVector.insert(myVector.begin() + 2, 25);
        for (auto& it : myVector) {
            std::cout << it << " ";
        }
        std::cout << std::endl;
        std::cout << "vector 删除末尾。" << std::endl;
        myVector.pop_back();
        for (auto& it : myVector) {
            std::cout << it << " ";
        }
        std::cout << std::endl;
        std::cout << "vector 删除中间。" << std::endl;
        myVector.erase(myVector.begin() + 3);
        for (auto& it : myVector) {
            std::cout << it << " ";
        }
        std::cout << std::endl;
```

```
        std::cout << "vector 查找。" << std::endl;
        auto iter = std::find(myVector.begin(), myVector.end(), 25);

        std::cout << "vector 第" << std::distance(myVector.begin(), iter)+1
            <<"个数据是"<< myVector[std::distance(myVector.begin(), iter)]
<< std::endl;

        return 0;
    }
```

五、 实验要求

1、 在自己电脑上正确安装 Visual Studio 软件.

2、 完成上述完整操作步骤

3、 复习 C++理论课程相关材料。